

# CLARAty Decision Layer Overview

Tara Estlin  
May 16, 2002

*CLARAty Decision Layer Team:* Tara Estlin and Caroline Chouinard

*CLEaR Team:* Forest Fisher, Dan Gaines, Steve Schaffer

*CMU Collaboration:* Reid Simmons

# Talk Organization

- Decision Layer summary and capabilities
- DL system architecture (as provided by CLEaR)
- Examples of DL domain models
- Code status and documentation

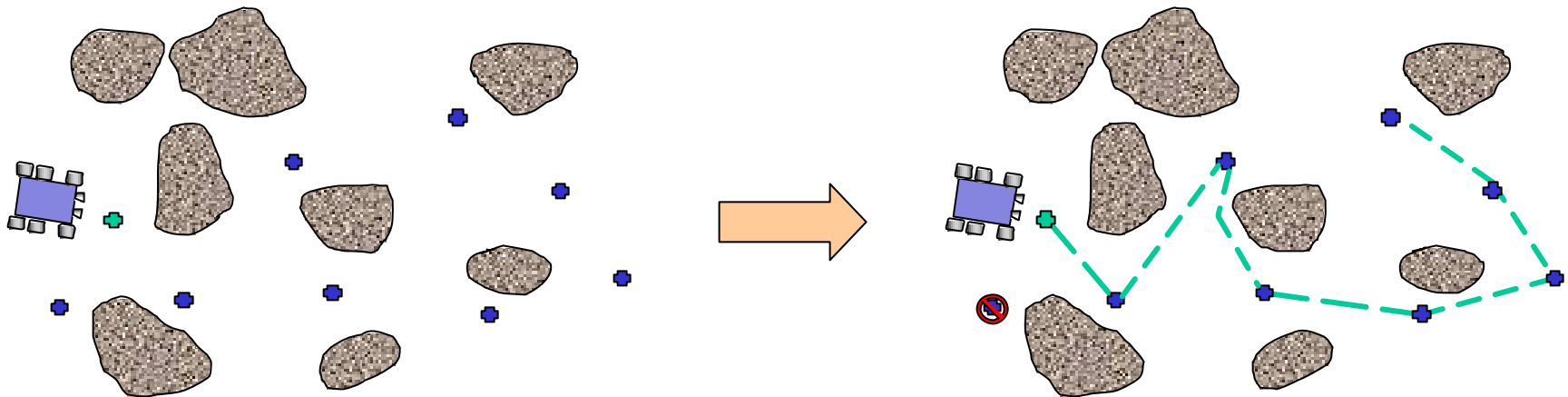
# Decision Layer Summary and Capabilities

# Decision Layer Summary

- Provides intelligent decision-making capabilities
- Designed to contain and enable integration of:
  - planning/scheduling techniques
  - executive techniques
  - other high-level autonomy techniques (e.g. data analysis)
- Also intended to provide flexible interface to CLARAty Functional Layer
- Current DL instantiation provided by CLEaR Framework
  - CLEaR: Closed Loop Execution and Recovery
  - Integrates CASPER dynamic planning system (JPL) with TDL executive (CMU)

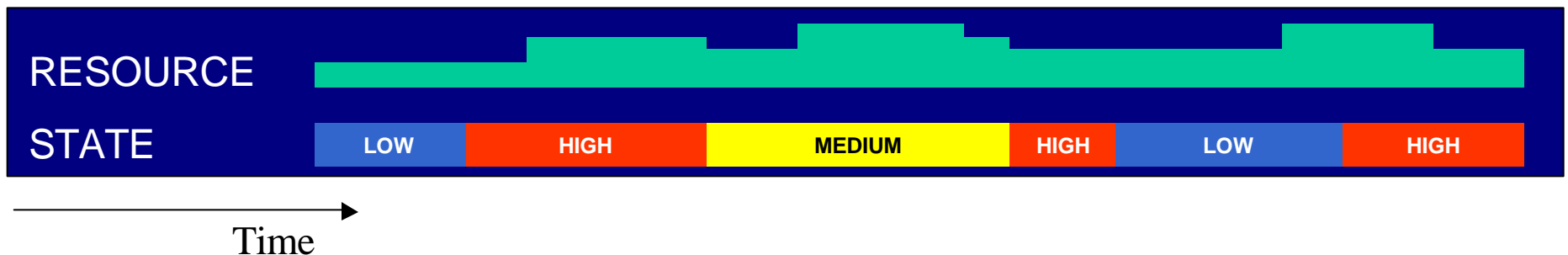
# Decision Layer Capabilities

- Autonomous rover-command generation to achieve high-level science and engineering goals
  - Given:
    - Initial state: Rover at position (x,y) w/ certain resource levels
    - Desired state (i.e., list of goals): Achieve science activities A, B and C
  - Produce:
    - Plan (i.e., list of commands) to make desired state true
    - E.g., navigate to loc A, raise mast, take image, lower mast, navigate to loc B, etc.



# Decision Layer Capabilities, cont.

- Reasoning about state, resource and temporal constraints
- Use of both declarative and procedural domain information
  - Activity X uses “20 Watts of Power” and requires “Camera to be on”
  - Activity A must be scheduled before Activity B within allowable range [10 sec, 30 sec]
  - Activity Z breaks down into sub-activities Z1, Z2 & Z3
  - If condition P holds then perform activity Q
- Timelines represent plan’s effects over time



# Decision Layer Capabilities, cont.

- Command execution and monitoring
  - Dispatch commands to FL and monitor relevant state and resource information
- Exception handling
  - Handle exceptions or failures
  - E.g., must retry rock grasp due to initial failure
  - E.g., motor overheats so disable arm
- Re-planning in light of changing context or goals
  - Modify global plan when conditions change
  - E.g., discard science target or traverse due to unexpected low power reserves
  - E.g., add science target due to unexpected opportunity

# Current Decision Layer Instantiation

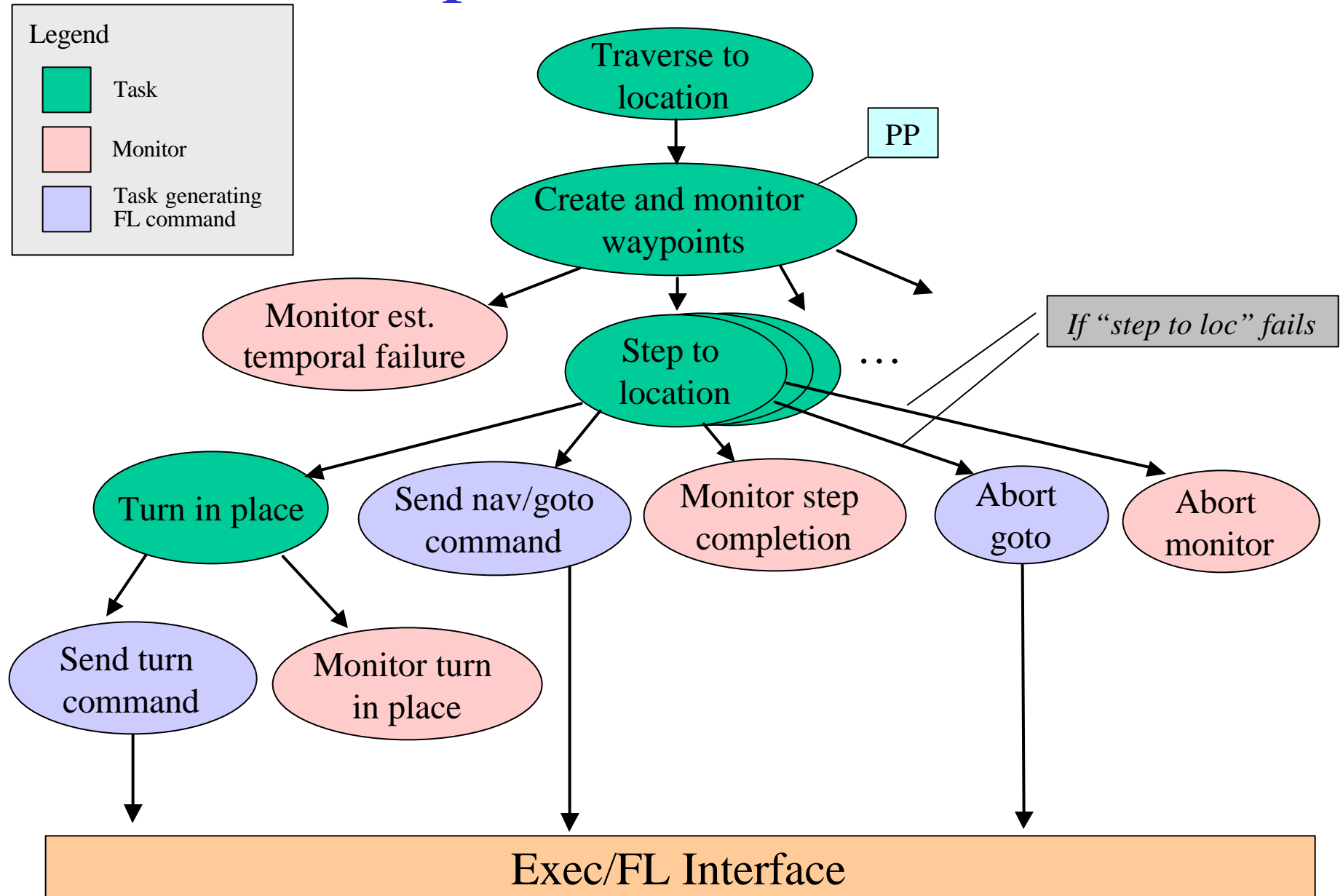
# CLEaR System

- CLEaR integrates two software components:
  - CASPER planning and scheduling system
    - Organized around activities
    - Reasons about resource, state and temporal constraints
    - Conflict detection
    - Global knowledge of plan and state/resource timelines
    - Provides:
      - Initial plan generation
      - Dynamic re-planning when state or goals changes
      - Declarative constructs for model information (e.g., activity pre-conditions and effects)
  - TDL executive
    - Organized around tasks and task trees
    - Reasons about task expansion based on current state
    - Event-driven decision-making
    - Provides:
      - Task decomposition and synchronization
      - Task execution and monitoring
      - Exception handling
      - Procedural constructs for model information (e.g., conditionals, iterative behavior)
- CLEaR task examining how planning and executive capabilities can be closely integrated to provide a more robust and responsive system

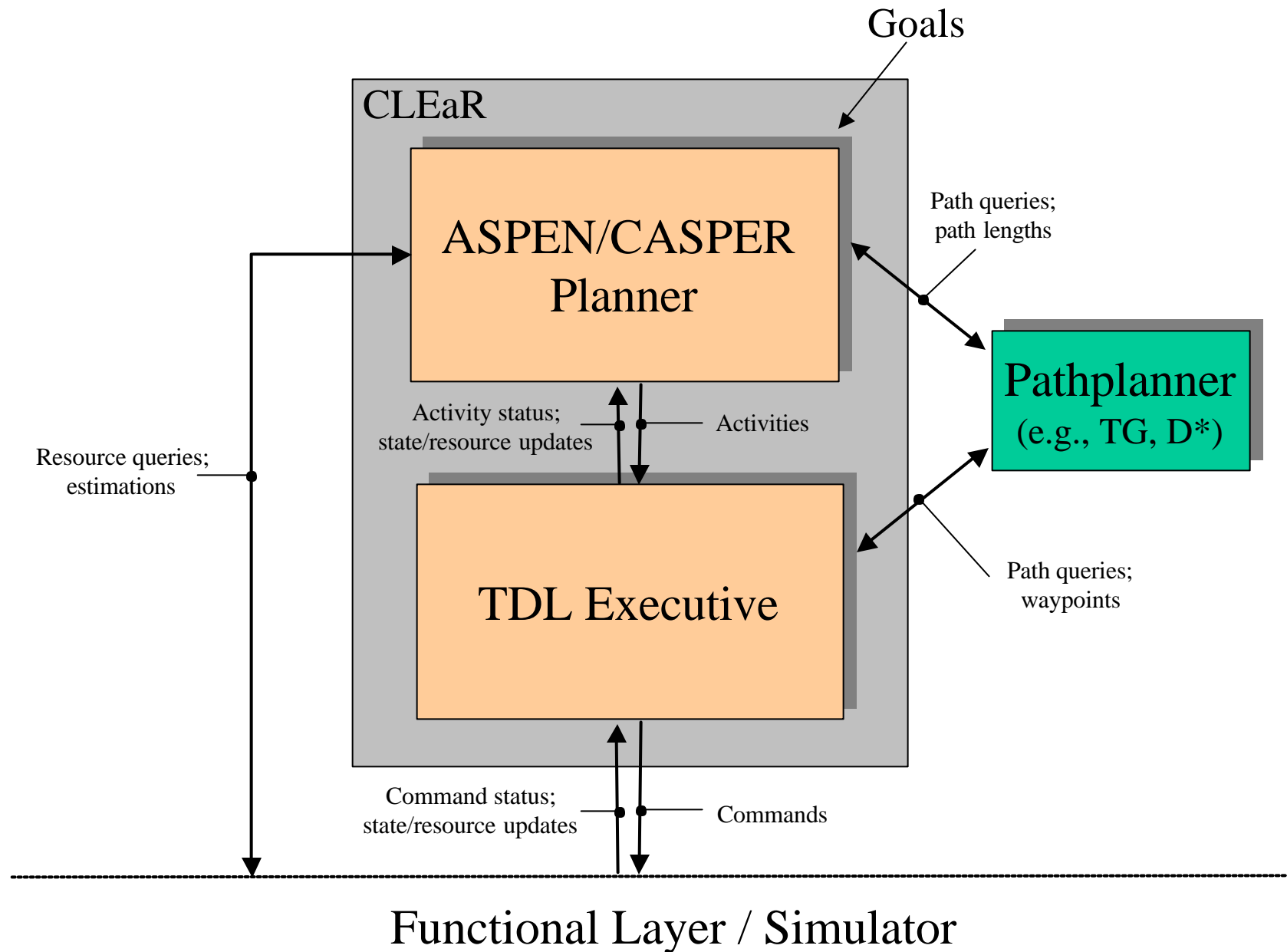
# Example ASPEN Plan



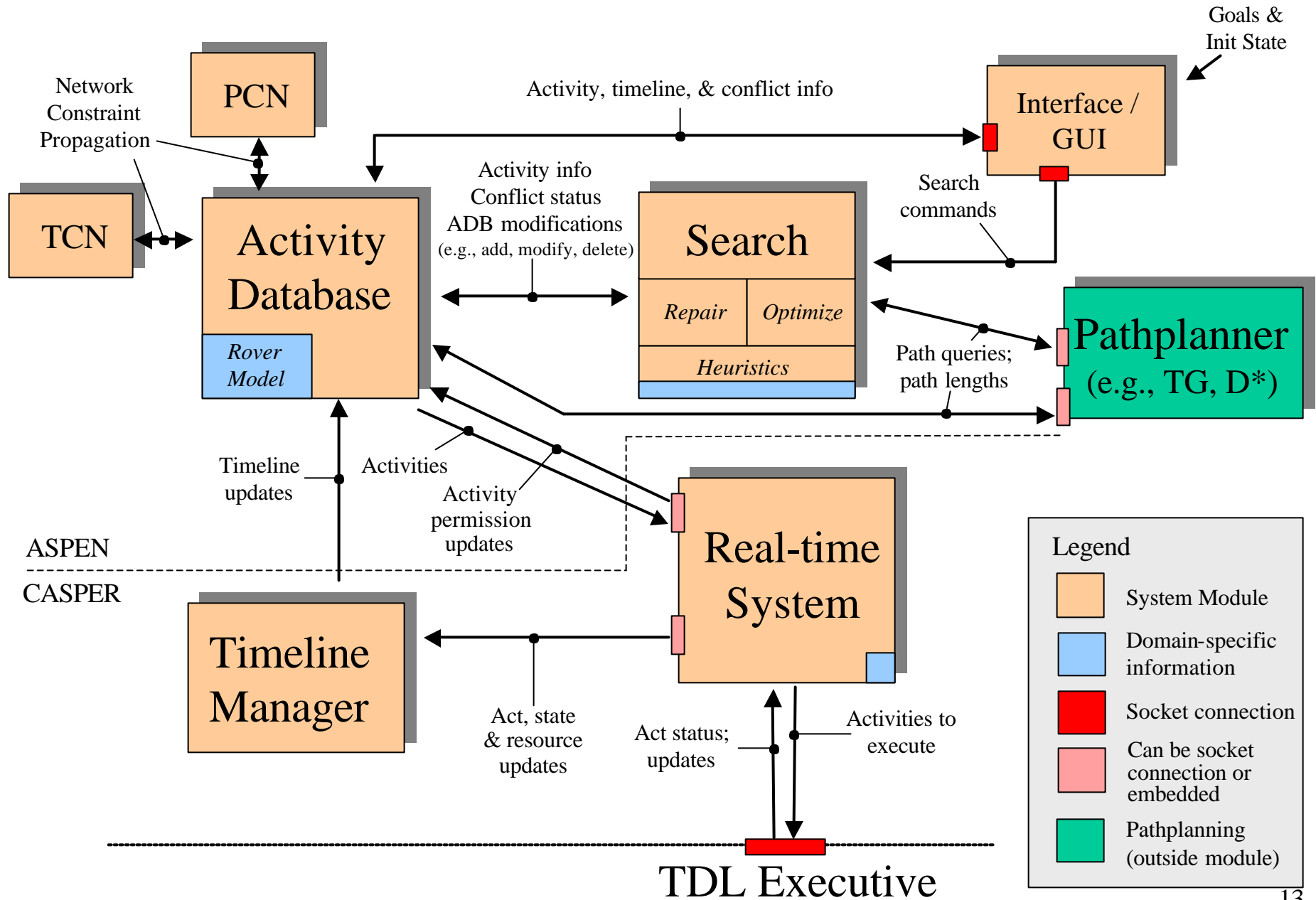
# Sample TDL Task Tree



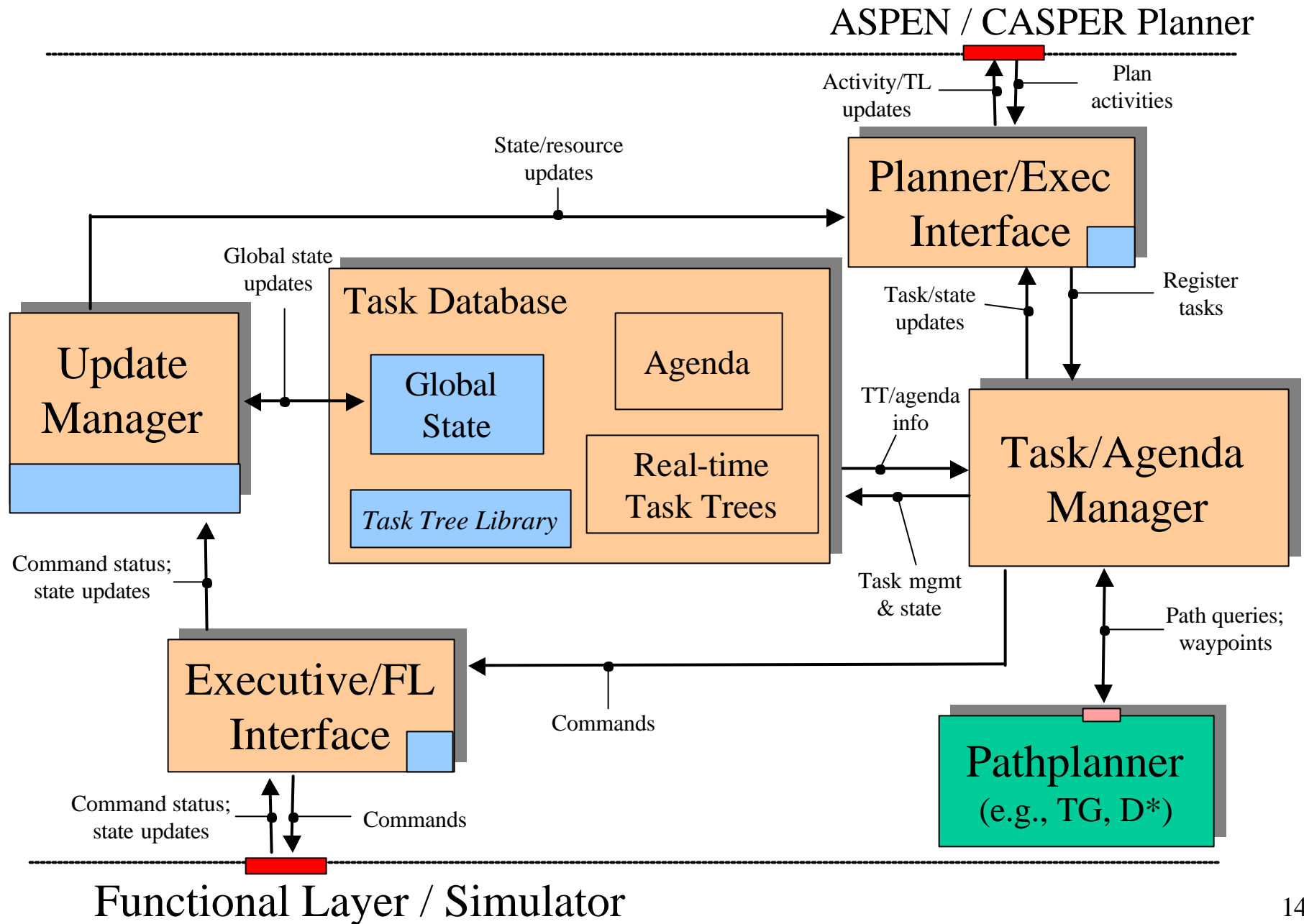
# CLEaR High-level View



# ASPEN/CASPER System Architecture



# TDL System Architecture



# Decision Layer Examples and Model Information

# Example ASPEN Plan



# ASPEN Model Organization

- Model Files
  - Activities.mdl
  - State-variables.mdl
  - Resources.mdl
- Init Files
  - Init-state.ini
  - Requests.ini (goals)
- Functions/code
  - Functions.cc (parameter functions)
  - Heuristic-functions.cc
  - User-search-functions.cc

# Sample ASPEN Activity Definition

```
Activity go_to_location {  
  position fromx, fromy, fromz, x, y, z;  
  angle fromheading, heading;  
  distance dist; // est. traverse distance  
  real speed = 1.655; // meters per min  
  real goto_power = 330.0; // Watts  
  int goto_energy; // Watt-hours  
  ...  
timeline_dependencies =  
  <fromx, fromy, fromz, fromheading> <- rover_orientation_sv;  
dependencies =  
  dist <- path_distance(fromx, fromy, x, y, pathplanner),  
  duration <- traverse_time(dist, speed, duration, ...),  
  goto_energy <- calculate_rover_energy(goto_power, duration);  
reservations =  
  day_night_sv must_be "day",  
  rover_energy_sv use goto_energy,  
  health_sv must_be "nominal",  
  rover_orientation_sv change_to <x,y,z,heading> at_end;  
};
```

# Sample ASPEN Resource & State Defs

```
Resource shovel { type = atomic; };
```

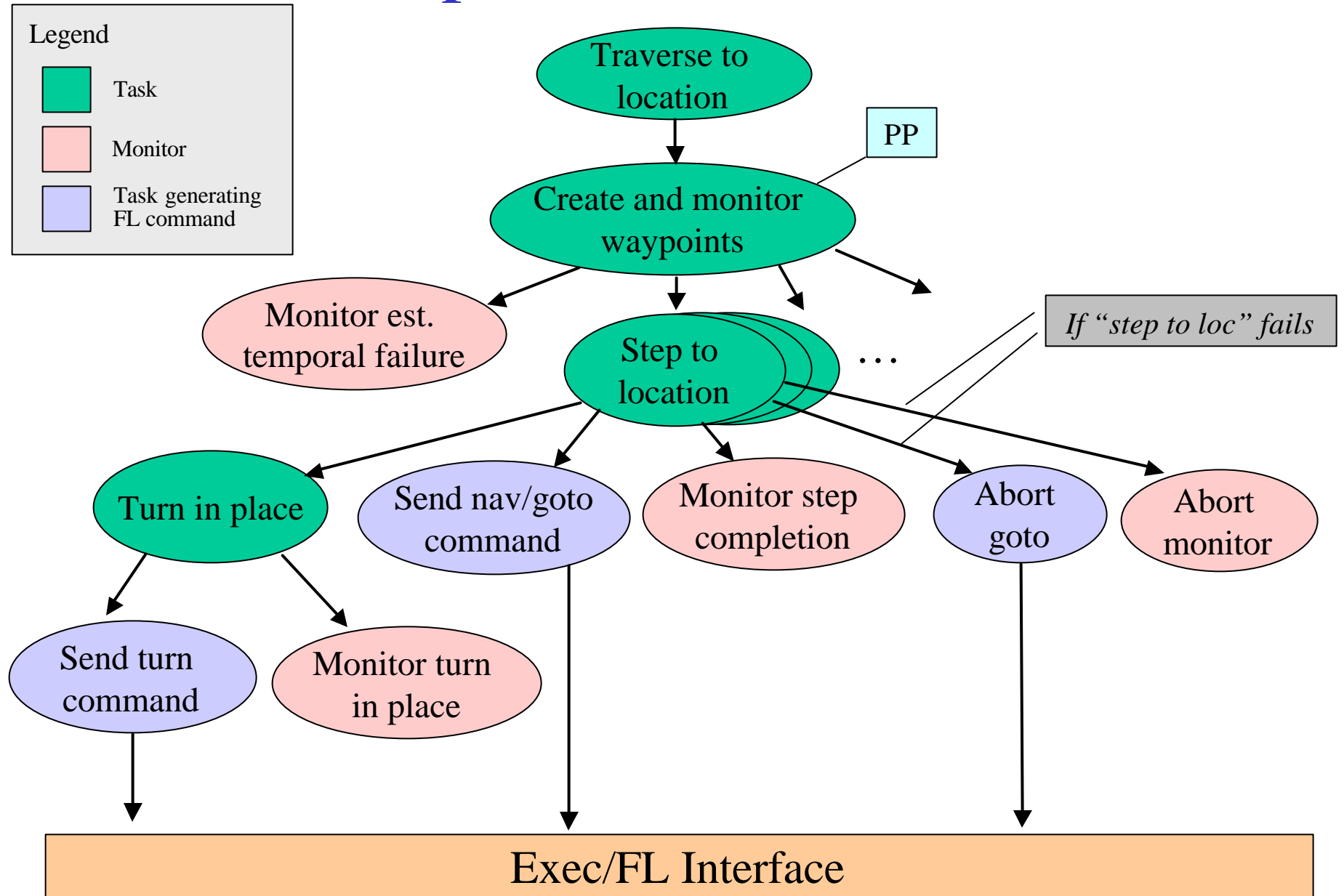
```
Resource spectrometer { type = atomic;  
    };
```

```
Resource battery {  
    type = depletable;  
    capacity = 69; // watts  
    min_value = 0;  
};
```

```
State_variable rover_orientation_sv {  
    // x, y, z, heading  
    states = <real, real, real, real>;  
    default_state = <0.0, 0.0, 0.0, 0.0>;  
}
```

```
State_variable mast_sv {  
    states = ("stowed", "deployed");  
    default_state = "stowed";  
};
```

# Sample TDL Task Tree



# TDL Model Organization

- Model information contained primarily in \*.tdl files
- Procedures in files similar to C++ functions
  - TDL based on top of C++ programming language
  - TDL modeling language provides extra constructs for task management (e.g., synchronization, monitoring, etc.)

# Sample TDL Task Definition

```
GOAL stepToLocation(TaskData *taskData, double destPosTol, int time_bound)
{
    float targetX, targetY, targetZ, targetHeading, priority;
    ...
    if( (tdlVars[TIME].getCurrentValue() < time_bound) &&
        (!tdlVars[TASK_TEMPORAL_FAILURE].getCurrentValue() ) )
    {
        Spawn turnInPlace(taskData,time_bound) With Wait;
        if(!commandMap[(int)(tdlVars[CUR_TRAVERSE_TID].getCurrentValue())].isSucceeded())
        {
            cout<<"stepToLocation::turnInPlace returned without completion. NOT proceeding with goto\n"<<flush;
            ...
        }
    }
    if( (tdlVars[TIME].getCurrentValue() < time_bound) &&
        (!tdlVars[TASK_TEMPORAL_FAILURE].getCurrentValue() ) )
    {
        tid = sendNavToCommand(targetX, targetY, targetZ, targetHeading, destPosTol, position_update_frequency);
        Spawn stepCompletion(taskData, tid, time_bound) With Wait;
        if(!commandMap[tid].isComplete())
        {
            Spawn abortCurrentGoTo(taskData) With Wait; //send an all-stop
            commandMap[tid].complete(FAILED);
            return;
        }
    }
    ...
}
```

# Sample TDL Monitor Definition

```
MONITOR stepCompletion(TaskData *taskData, int tid, int time_bound)
serial, period 0:0:2.0, maximum trigger 1
{
  if(commandMap[tid].isComplete())
  {
    cout<<"stepCompletion: TRIGGER (task compl)\n"<<flush;
    TRIGGER();
  }
  if(tdlVars[TASK_TEMPORAL_FAILURE].getCurrentValue())
  {
    cout<<"stepCompletion: TRIGGER (temporal failure)\n"<<flush;
    TRIGGER();
  }
  if(tdlVars[TIME].getCurrentValue() >= time_bound)
  {
    cout<<"stepCompletion: TRIGGER (time bound = ... \n"<<flush;
    TRIGGER();
  }
  ...
}
```

# Decision Layer Code Status and Documentation

# Status of Code

- System requirements
  - Currently developed using Sparcworks compiler, Solaris 2.6 & 2.8
  - Have created working binary for Solaris 2.7 (telerobotics sparcs)
  - Currently porting code to g++ under Linux (TDL already in Linux)
- CLARAty Repository
  - Have not checked in yet
  - Holding on paperwork and porting code
- Model information
  - Have ASPEN and TDL models based on FY01 scenario
    - Work for both R7 & R8 with small parameter changes
    - Currently maintain two separate models for ASPEN & TDL
  - Model(s) primarily reason about traverses, comm and science activities, and power and memory resources
  - DL designed to receives updates from FL for:
    - Position, memory & energy
    - Currently extending to receive map updates
  - Model(s) will need to be extended for additional scenarios and/or as new rover components are added or tested (e.g., arm, mast, comm)
    - System source code may require extensions as well (e.g., optimization)

# Documentation

- Some documentation exists on CLARAty DL web page
  - <http://claraty/Development/DL%20Documentation/index.html>
  - Will be extending this to have step-by-step instructions for running FY01 (and future) scenarios in simulation and on rovers
- ASPEN/CASPER documentation
  - <http://www-aig.jpl.nasa.gov/public/planning/aspen/>
  - <http://www-aig.jpl.nasa.gov/public/planning/casper/>
  - ASPEN User's Manual can be found at:
    - <http://www-aig.jpl.nasa.gov/public/planning/aspen/usersguide.pdf>
- TDL documentation
  - <http://www-2.cs.cmu.edu/~tdl/>
  - TDL Quick Reference Manual can be found at:
    - <http://www-2.cs.cmu.edu/~tdl/tdl.html>